استاد فیلی — تکلیف سری ۴ DS — ۸۱۰۱۰۰۰۸۴ — محمد امانلو

---

۱)

الف)

$h(12,0) = [12+0] \% 12 = 0$

$h(21,0) = [21+0] \% 12 = 9$

$h(25,0) = 1$

$h(9,0) = 9 \rightarrow h(10,0) = 10$

$h(14,0) = 1$

$h(14,0) = 2$

$\rightarrow$

12	25	14	⋮	21	9
0	1	2		9	10

ب)

$h_1(12) = 0$	$h_2(12) = 1$	$\xrightarrow{i=0}$	ans = 0
$h_1(21) = 9$	$h_2(21) = 2$	$\xrightarrow{''}$	9
$h_1(25) = 1$	$h_2(25) = 2$	$\xrightarrow{''}$	1
$h_1(9) = 9$	$h_2(9) = 2$	$\xrightarrow{i=1}$	11
$h_1(14) = 1$	$h_2(14) = 2$	$\xrightarrow{''}$	3

$\rightarrow$

12	25	⋮	14	⋮	21	⋮	9
0	1		3		9		11

---

۲) الف) ابتدا با Merge sort آرایه را از مرتبه $O(n\log n)$ مرتب می‌کنیم. سپس با استفاده از Binary search برای هر عنصر مقدار متناظر را یافت می‌کنیم. یا در واقع برای هر عنصر از نیمه به دنبال عنصر متناظر می‌گردیم در صورتی که مقدار کمتری نیاز بود به نیمه سمت چپ و در صورتی که به مقدار بیشتری نیاز بود به نیمه راستی می‌رویم که این روش نیز از $O(n\log n)$ است و:

$O(n\log n) + O(n\log n) = 2O(n\log n) = O(n\log n)$

ب) با استفاده از Hash Table که اگر عنصر از $s/2$ کمتر باشد، $h(n)=n$ و اگر بیشتر باشد آن را با $(h(n)=s/2)$ جمع می‌کنیم. هر خانه را بررسی می‌کنیم تنها عنصرهای یک خانه هستند که می‌توانند مجموع s داشته باشند سپس با بررسی خانه‌ها از مرتبه n، جفت مورد نظر را پیدا می‌کنیم.

dn

۳) برای جمع مقادیر با مجموع اعداد آرایه از صفر تا n را حساب می‌کنیم بدین صورت که:

S[1] = a[1]
S[2] = S[1] + a[2]
S[3] = S[2] + a[3]

که این عمل به علت یکبار حرکت روی آرایه از O(n) است. در واقع که هر مرحله مقدار موجود a(i) را با مقدار متغیر S جمع کرده و در S(i) می‌ریزیم. سپس به ازای i ها و j های موجود S(i) و S(j) را وارد یک جدول درهم‌سازی می‌کنیم.

حال با O(n) روی تمام های موجود می‌گردیم تا جفت‌هایی که S(i) آن‌ها با S(j) برابر باشند بدین معنی که مجموع اعداد مابین صفر است را انتخاب کرده و مقدار j تا i را برمی‌گردانیم!

$$O(n) + O(n) = \boxed{O(n)}$$

---

۴) در این روش که مبتنی بر مقایسه نیست و از شمارش عناصر استفاده می‌کند. ابتدا یک آرایه به طول بزرگترین عنصر آرایه به اضافه یک می‌سازیم و همه‌ی مقادیر آن را صفر می‌کنیم. (اسم این آرایه B است) سپس با چرخش روی آرایه‌ی اول به ازای هر یک از مقادیر در آرایه‌ی اول خانه‌ای با اندیس مقدار این خانه به اضافه ۱ رفته و به مقدار قبلی آن در B یکی اضافه می‌کنیم. و همین کار را تا آخر تکرار می‌کنیم و سپس هر عنصر B را با عناصر قبلی جمع می‌کنیم. در نهایت یک آرایه هم طول آرایه اول ایجاد کرده (C) از آخرین خانه‌ی آرایه‌ی اول شروع کرده و به ابتدای آن می‌رویم هر بار خانه‌ی انتهایی از آرایه اول را در خانه‌ی یکی بیشتر از مقدارش در آرایه B مشخص کرده و می‌رویم و یکی از آن کم می‌کنیم. و سپس به خانه‌ی یکی کمتر از مقدارش رفته و مقدار را در آن در C قرار می‌دهیم.

برای آرایه [۹,۲,۶,۹,۹,۲,۴,۸,۴] مثلا:

هر کدام از اعداد را شمارش را در اندیس مربوط نوشتیم

① B_1 = [۰,۰,۲,۰,۲,۰,۱,۰,۱,۳]
B_2 = [۰,۰,۲,۲,۴,۴,۵,۵,۶,۹]

② C = [ـ,ـ,ـ,ـ,ـ,ـ,ـ,ـ,۹]　　B = [۰,۰,۲,۲,۴,۴,۵,۵,۶,۸]

③ C = [ـ,۲,ـ,ـ,ـ,ـ,ـ,ـ,۹]　　B = [۰,۰,۱,۲,۴,۴,۵,۵,۶,۸]

④ C = [ـ,۲,ـ,ـ,ـ,۶,ـ,ـ,۹]　　B = [۰,۰,۲,۲,۴,۴,۴,۵,۶,۸]

⋮　　⋮

$$\boxed{C = [۲,۲,۴,۴,۶,۸,۸,۹,۹]}$$

۵) ابتدا یک tuple شامل بازه‌ی شروع و پایان می‌سازیم و آن‌ها را براساس سمت پایان جلسات sort می‌کنیم.

سپس ابتدای هر جلسه را با انتهای جلسه‌ی قبلی مقایسه می‌کنیم اگر بیشتر بود مشکلی نیست اما اگر کمتر بود تداخل وجود داشته و این تداخل تا یک سمت جلسه بود، جلسه‌ی کوتاه‌تر را نگه داشته و جلسه‌ی بلندتر را حذف می‌کنیم اما اگر جلسه‌ای هم با بعدی هم با قبلی تداخل داشت و جلسه‌ی بعدی و قبلی تداخل نداشتند جلسه‌ای که بیشترین تداخل را دارد حذف می‌کنیم. در واقع هر بار با انتساب هر جلسه با باینری سرچ (Binary search) جلسات تداخل‌دار یافت می‌شود و ابتدا پرتداخل‌ترین و سپس زمان‌برترین جلسه‌ی تداخل‌دار حذف می‌شود چون الگوریتم یک بار حرکتش روی داده‌ها و یک BS دارد پس از مرتبه O(nlogn) است!

---

۶) در ابتدا آرایه را با استفاده از Mergesort با O(n logn) مرتب می‌کنیم. با تفاوت در اینکه در هنگام بازگشت در Merge sort وقتی عدد اول در راست با اولین عدد در چپ مقایسه می‌شود اگر عدد سمت چپ بزرگتر باشد، به طول سمت چپی تعداد وارونگی‌ها است. سپس عدد بعدی سمت راست مقایسه شده و باز مقدار وارونگی جدید با مقدار قبلی جمع می‌شود.
این کار برای تمامی اعداد راست انجام شده و در نهایت تعداد وارونگی‌ها بدست می‌آید. این کار در تعداد اجرای حلقه‌های کلی Mergesort تغییری ایجاد نمی‌کند و در عین انجام آن صورت می‌پذیرد و تغییری در عملیات‌های اصلی داخل Merging پس از همان مرتبه Mergesort یعنی O(n logn) می‌باشد!

Finish!